UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	DRAWINGS	TOT CLAIMS	IND CLAIMS
10/164,260	06/05/2002 ✓	2152	1370	60001.0182US01 ✓	6	55	3

CONFIRMATION NO. 1092

27488
MERCHANT & GOULD
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

CL

FILING RECEIPT

OC000000008437455

Date Mailed: 07/11/2002

Receipt is acknowledged of this nonprovisional Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Filing Receipt Corrections, facsimile number 703-746-9195. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).**

Applicant(s) ✓

Jeff Reynar, Woodinville, WA;
Paul Broman, Renton, WA;
Brian Jones, Redmond, WA;
Robert Little, Redmond, WA;

Assignment For Published Patent Application ✓

Microsoft Corporation;

Domestic Priority data as claimed by applicant

Foreign Applications

If Required, Foreign Filing License Granted 07/11/2002

Projected Publication Date: 12/11/2003

Non-Publication Request: No ✓

Early Publication Request: No

Title

Mechanism for downloading software components from a remote source for use by a local software application ✓

D✓

Preliminary Class
709

LICENSE FOR FOREIGN FILING UNDER
Title 35, United States Code, Section 184
Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Office of Export Administration, Department of Commerce (15 CFR 370.10 (j)); the Office of Foreign Assets Control, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

M&G No. 60001.0182US01/MS 300077.1

MECHANISM FOR DOWNLOADING SOFTWARE COMPONENTS FROM A REMOTE SOURCE FOR USE BY A LOCAL SOFTWARE APPLICATION

Technical Field

This invention relates to a mechanism for downloading software components from a remote source for use by a software application.

Background of the Invention

Computer software applications allow users to create a variety of documents to assist them in work, education, and leisure. For example, popular word processing applications allow users to create letters, articles, books, memoranda, and the like. Spreadsheet applications allow users to store, manipulate, print, and display a variety of alpha-numeric data. Such applications have a number of well-known strengths, including rich editing, formatting, printing, calculation, and offline editing.

Most modern computer software applications do not contain all necessary programming for providing the functionality of the software application at application boot-up time. Most computer software applications are associated with a number of separate components or individual modular routines that have been compiled and that have been dynamically linked to the software application, and are ready to use with the software application or with other components of the software application upon being "called" by the software application. Such components take a variety of forms, including components known as dynamic-link libraries (dll), which are executable files or routines stored separately from the software application and which are loaded by the software application only when they are needed.

Various software development models have been developed for using individual modular software routines, such as the so-called component object model or COM, which allows for the building of the software application from individual modular

routines that may be plugged in, or unplugged, to a larger software application when needed.

Such systems allow for individual modular software routines to be held in storage until they are needed so that they do not unnecessarily consume memory and processing time. Other advantages of such software systems include the ability of software developers to make changes or corrections to individual components without affecting the overall programs to which they are associated. Moreover, using a component object modeling system for building software applications allows software developers to use individual software components in multiple software applications.

An example of an individual component includes a component responsible for drawing a graphical user interface, such as a toolbar, that may be used for drawing the same toolbar on a number of software applications. Other examples include Extensible Markup Language (XML) programs, Extensible Markup Language transformation programs, or any sub-routine that may be downloaded to, and associated with, a software application for which it may provide some type of support or functionality.

A typical implementation of a software application using a variety of modular software routines or components requires that a software application, along with all required components, be shipped to the end user for downloading on to the end user's local computer. If the software application developer later finds an error in the programming of an individual component, typically the user must receive a repair or "patch" from the software developer that must be installed on the end user's computer to repair the error in the programming of the component. Unfortunately, the repair or "patch" must be sent to every user who has purchased, downloaded, and/or installed the software application using the defective component.

Another exemplary implementation of software applications using modular software routines include documents that are enabled to use the functionality of those modular routines to add special or "smart" functionality to the documents. For example, a user may need an additional software component or set of components to add help content or additional document actions to the existing functionality of the application being used to create and edit the document. Unfortunately, adding

additional components or sets of components requires the user to receive an installation package and install the new components in order to add the new or enhanced functionality.

It is with respect to these and other considerations that the present invention has been made.

Summary of the Invention

The present invention provides a method and system for downloading software components from a remote source to a software application for providing updates or additions to the application's functionality. Generally described, a schema is attached to a document defining permissible data content, data type and data structure from the document. The document is structured to associate the document with the schema. A solution is associated with the document structure. For example, a solution associated with a document structure may include help tools associated with an expense report document structure, or the expense report structure may include a cost section (structure) associated with a currency conversion solution. A plurality of software components comprising the solution is assembled at a location remote from the document. The document is enabled to call the solution to obtain functionality provided by the plurality of software components. If it is determined that the plurality of software components are required by the application to provide functionality to the document, the plurality of software components are downloaded to the application. Before downloading the software components to the application the software application is connected to the remote server. The connection may be via a distributed computing environment such as the Internet, an Intranet, or via a local area network or wireline or wireless telecommunications network.

A solution property is attached to the document for pointing the document to the solution, and a solution location is attached to the document for enabling the document to locate the solution at the remote location. The solution and the plurality of software components comprising the solution are listed in a manifest of software components.

The manifest is stored in a remote library of software components on a remote server accessible by the document.

Prior to downloading the software components to the application, a determination is made as to whether the plurality of software components for providing the solution is present in a local library of software components. If the plurality of software components is not present in the local library of software components, the manifest at the remote library of software components is called for obtaining the solution. Prior to downloading the software components to the application, a query may be presented to the user to determine whether the plurality of software components should be downloaded. The query may include a notification that the software components have been updated at the remote server, or that additional software components have been added to the plurality of software components. Additionally, the user may receive notification that a set amount of time has elapsed since the last download of software components to the application.

Prior to downloading the software components to the document, the security of the manifest is checked. The security of the manifest is checked by determining whether the manifest is located at a trusted site or by determining whether the manifest is located from a trusted intranet site. Additionally, security of the manifest may be determined by checking digital signatures applied to files contained on the manifest by the creator or administrator of those files against a list of trusted digital signatures. In either case, if the security of the manifest cannot be assured, downloading the plurality of software components to the application is prevented. Alternatively, after downloading the plurality of software components to the application, but prior to installing those components for use by the application and document, a checksum value may be obtained representing the contents of the manifest at the remote server site, and that checksum value may be compared against a second checksum value representing the contents of the manifest after the contents have been downloaded. If the second checksum value differs from the first checksum value, installing the software components for use by the application is prevented.

After checking the security of the manifest, the contents of the manifest are compared to software components present in a local library to determine whether updates or additions to the contents of the manifest have been made. If updates or additions to the contents of the manifest have been made, downloading of the contents from the remote server for use by the application may proceed. After the software components are downloaded to the application, those components are installed for providing to the application and document the functionality associated with the downloaded software components.

According to another aspect of the invention, an Extensible Markup Language (XML) schema is attached to the document to define permissible data content, data types and data structures for the document. Particular portions of the document are annotated with XML structure associated with the schema. A solution comprising a plurality of software components is associated with XML elements making up the XML structure. The document is enabled to call the solution to obtain functionality provided by the plurality of software components upon initiation of editing the document within an XML element associated with a particular solution.

Once a document is open and a computer cursor is placed within a particular portion of the document, a list of XML elements is generated enclosing the position of the cursor. A determination is made as to whether a solution property pointing the document to the solution is associated with the list of XML elements. A determination is made as to whether the plurality of software components for providing the solution is present in a local library of software components, and if the plurality of software components is not present in the local library of software components, the manifest at the remote library is called for obtaining the solution.

These and other features, advantages, and aspects of the present invention may be more clearly understood and appreciated from a review of the following detailed descriptions of the disclosed embodiments and by reference to the appended drawings and claims.

Brief Description of the Drawings

Fig. 1 is a simplified block diagram illustrating a client-server architecture for use in conjunction with an embodiment of the present invention.

Fig. 2 is a block diagram of a computing system and associated peripherals and network devices that provide an exemplary operating environment for the present invention.

Fig. 3 is a computer screen display of a software application graphical user interface through which is displayed a document and associated contextually sensitive actions and help content according to an embodiment of the present invention.

Fig. 4 is a flow chart illustrating a method for downloading components required for operating and for providing functionality to a client-side software application.

Fig. 5 illustrates a computer-generated dialog box for offering multiple document solutions to the user.

Fig. 6 illustrates a computer-generated dialog box for assisting a user with downloading components required by a software application.

Fig. 7 illustrates a computer-generated dialog box for prompting a user of the need to connect to the internet for downloading components according to an embodiment of the present invention.

Fig. 8 illustrates a computer-generated dialog box for warning a user of a potential security risk associated with downloading components to the user's computer.

Fig. 9 illustrates a computer-generated dialog box for alerting a user of a problem associated with downloading components required by the user's software application.

Detailed Description

Referring now to the drawings in which like numerals refer to like parts or components through the several Figures. Fig. 1 is a simplified block diagram illustrating a client-server architecture for use in conjunction with an embodiment of the present invention. All components and files necessary to operate or fully implement the functionality or solutions available to a software application 110 are identified and are

assembled on a local library of software components in the schema library 105. For a detailed description of the schema library 105, see U.S. Patent Application entitled "*Schema Library,*" Serial No. _____, filed _____, which is incorporated herein by this reference as if fully set out herein.

All components and files that may be utilized to update or add to functionality available to the application 100 are identified and are assembled on a manifest 38 which may be located in a remote library of software components on a remote web server 49 accessible by the user via the user's computer 20. If the user is informed that the components on her client-side computer 20 should be updated, or that corrections or improvements to existing components are available, or that new functionality is available that will transform the user's existing application 100 and document 110 into a "smart" application and "smart" document, the user may connect to the web server 49 via the Internet 51, or via an intranet 52, to download the required components. Alternatively, the user may connect to the web server 49 via any suitable distributed computing environment, including wireline and wireless telecommunications networks, for allowing the user to connect to the web server 49 to download files from the manifest 38.

The manifest 38 may include all components, including dlls, component add-ins, Extensible Markup Language (XML), schema files and all associated XML files required by a software application for operating properly or required for improving, or adding, functionality to the software application 100. According to an embodiment of the present invention, the manifest 38 may also include information helpful to the end user, or to the end user's computer, for installing the downloaded components. For example, the manifest 38 may include information specifying the type for a given dynamic-link library (dll) so that the client-side computer 20 can properly register the dll including any need for specific registry keys for properly registering the dll.

A schema may be attached to the manifest 38 to define permissible data content, data type and data structure of the manifest as a file and for associating the manifest with files, documents and applications that may call the manifest 38 for obtaining a

download of required components. For purposes of example only and not for limitation of the invention described herein, the following is an example XML-based schema that may be attached to the manifest 38 for associating the manifest with files, documents and applications that may call the manifest 38 for obtaining a download of required components.

Sample Manifest Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:schemas-microsoft-com:smartdocuments:manifest"
    targetNamespace="urn:schemas-microsoft-
com:smartdocuments:manifest" elementFormDefault="qualified">
  <xsd:annotation>
   <xsd:documentation xml:lang="en">
    The schema for office smart document, transform and schema manifests.
   </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="manifest" type="manifestType" />
  <xsd:complexType name="manifestType" mixed="true">
    <xsd:sequence>
      <xsd:element name="version" type="xsd:string" />
      <xsd:element name="location" type="xsd:string" />
      <xsd:element name="updateFrequency"
type="zeroAndPositiveInteger" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="uri" type="xsd:string" />
      <xsd:element name="manifestURL" type="xsd:string"
minOccurs="0" maxOccurs="1" />
      <xsd:element name="solution" type="solutionDefn"
minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="solutionDefn">
    <xsd:sequence>
      <xsd:element name="solutionID" type="xsd:string" />
```

```
                    <xsd:element name="type" type="solutionType" />
                    <xsd:element name="alias">
                      <xsd:complexType>
                        <xsd:simpleContent>
                          <xsd:extension base="xsd:string">
                          <xsd:attribute name="lcid" type="xsd:string"
use="required"/>
                          </xsd:extension>
                        </xsd:simpleContent>
                      </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="solutionSpecific" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                    <xsd:element name="file" type="fileType" minOccurs="1"
maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:simpleType name="solutionType">
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="schema" />
                    <xsd:enumeration value="transform" />
                    <xsd:enumeration value="smartDocument" />
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:complexType name="fileType">
                <xsd:all>
                    <xsd:element name="runFromServer" type="xsd:string"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="type" type="typeDefn" />
                    <xsd:element name="application" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                    <xsd:element name="version" type="xsd:string" />
                    <xsd:element name="context" type="xsd:string" minOccurs="0"
maxOccurs="1" />
```

```
                    <xsd:element name="filePath" type="xsd:string" />
                    <xsd:element name="installPath" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                    <xsd:element name="register" type="xsd:string" minOccurs="0"
maxOccurs="1" />
                    <xsd:element name="CLSID" type="xsd:string" minOccurs="0"
maxOccurs="1" />
                    <xsd:element name="progID" type="xsd:string" minOccurs="0"
maxOccurs="1" />
                    <xsd:element name="regsvr32" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                    <xsd:element name="regasm" type="xsd:string" minOccurs="0"
maxOccurs="1" />
                    <xsd:element name="registry" type="registryType"
minOccurs="0" maxOccurs="1"/>
                </xsd:all>
            </xsd:complexType>
            <xsd:simpleType name="typeDefn">
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="template" />
                    <xsd:enumeration value="smartTagList" />
                    <xsd:enumeration value="solutionList" />
                    <xsd:enumeration value="schema" />
                    <xsd:enumeration value="transform" />
                    <xsd:enumeration value="actionHandler" />
                    <xsd:enumeration value="solutionActionHandler" />
                    <xsd:enumeration value="recognizer" />
                    <xsd:enumeration value="solutionRecognizer" />
                    <xsd:enumeration value="solutionList" />
                    <xsd:enumeration value="COMAddIn" />
                    <xsd:enumeration value="assembly" />
                    <xsd:enumeration value="XMLFile" />
                    <xsd:enumeration value="installPackage" />
                </xsd:restriction>
```

```
        </xsd:simpleType>
        <xsd:complexType name="registryType">
            <xsd:all>
                <xsd:element name="registryKey" type="registryKeyType"
5   minOccurs="1" />
            </xsd:all>
        </xsd:complexType>
        <xsd:complexType name="registryKeyType">
            <xsd:all>
10              <xsd:element name="keyName" type="xsd:string" />
                <xsd:element name="keyValue" type="keyValueType" />
            </xsd:all>
        </xsd:complexType>
        <xsd:complexType name="keyValueType">
15          <xsd:all>
                <xsd:element name="valueName" type="xsd:string" />
                <xsd:element name="valueType" type="registryValueType" />
                <xsd:element name="value" type="xsd:string" />
            </xsd:all>
20      </xsd:complexType>
        <xsd:simpleType name="registryValueType">
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="REG_SZ" />
                <xsd:enumeration value="REG_EXPAND_SZ" />
25              <xsd:enumeration value="REG_DWORD" />
            </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="zeroAndPositiveInteger">
            <xsd:restriction base="xsd:integer">
30              <xsd:minInclusive value="0"/>
            </xsd:restriction>
        </xsd:simpleType>
        </xsd:schema>
```

If the user has opened a software application 100 that requires the functionality of various components to transform the user's document 110 into a "smart" document, as described below, the mechanism of the present invention may be used to download all components required for that operation. In that case, an XML schema that describes the type(s) of files and data structures that together make up a solution to transform the user's document 110 into a "smart" document, as well as information about registering those components and installing them on the user's computer 20 may be stored on the manifest 38.

For example, if the user receives a document, such as a template, for the preparation of a resumé that "points" to a solution for providing helpful information, and actions, associated with completing the text and data fields of the template, the template or document received by the user may point to a remotely stored manifest 38 so that the user may download all files necessary for fully implementing the solution referred to by the document received by the user. For a detailed description of a method and system for creating, implementing and using "smart" documents such as the document 110, illustrated in Fig. 3, see U.S. Patent Application entitled "*Providing Contextually Sensitive Actions and Help Content In Computer-Generated Documents,*" Serial No. _____, filed _____, which is incorporated herein by this reference as if fully set out herein.

Once the manifest 38 is created, and all of the necessary files are listed to implement a given solution, or to correct or improve the performance of a given application, a reference to the solution or updates may be made in the document, such as a word processing document, or spreadsheet document utilized by the user on the client-side computer 20. The manifest of files and solutions may be referred to by a solution or manifest ID to allow the user's client-side application and documents to point to and obtain information from, the remote manifest. The solution ID associated with software components pointed to by the document or application are stored in a solution property 115 attached to the document 110.

The location of the manifest 38, including required components and desired solutions, is referred to according to the solution location 118. The solution

location 118 may include the URL of the manifest 38 on the remote web server 49. If the user only has the document, as in the case where the user has received the document from a friend via electronic mail, the application 100 may call the web server 49 via the solution location 118, and by utilizing the solution ID from the solution property 115, the application may obtain the manifest 38 to determine what components must be downloaded, installed, registered, and implemented to provide the user with required or desired functionality. Now, the user has the document 110 and a set of installed "ready to run" files and other components that the software document 110 needs to enable the proper operation of the software application 100 or to enable the application 100 to transform the document 110 into a "smart" document. Advantageously, the requirement to receive an installation package from the software developer containing software repairs or "patches" or containing necessary functionality to improve the performance of original functionality of the application is avoided.

Fig. 2 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of an application program that runs on an operating system in conjunction with a personal computer, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, cell phones, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to Fig. 2, an exemplary system for implementing the invention includes a conventional personal computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that couples the system memory to the processing unit 21. The system memory 22 includes read-only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 further includes a hard disk drive 27, a magnetic disk drive 28, e.g., to read from or write to a removable disk 29, and an optical disk drive 30, e.g., for reading a CD-ROM disk 31 or to read from or write to other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide non-volatile storage for the personal computer 20. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD-ROM disk, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 25, including an operating system 35, one or more application programs 100, a word processor program module 37 (or other type of program module), program data, such as the manifest 38, and other program modules (not shown).

A user may enter commands and information into the personal computer 20 through a keyboard 40 and pointing device, such as a mouse 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via

an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers or printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the personal computer 20, although only a memory storage device 50 has been illustrated in Fig. 2. The logical connections depicted in Fig. 2 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the LAN 51 through a network interface 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the WAN 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Fig. 3 is a computer screen display of a software application graphical user interface through which is displayed a document and associated contextually sensitive actions and help content according to an embodiment of the present invention. The document 110 illustrated in Fig. 3 represents a "smart" document that may point to functionality that may be downloaded from the manifest 38. The document 110 illustrated in Fig. 3 is described for purposes of example only and is not limiting of the invention described herein. The word processing application 100 provides typical functionality associated with a word processor accessible via drop down menus such as,

File, Edit, View, Insert, Format, etc. The document 110 is displayed in the work area of the application 100, and a document actions pane 135 is illustrated to the right of the document 110.

When the user places her computer cursor within a particular section of the document 110, for example the "objectives" section 125 illustrated in Fig. 3, the user is provided with actions and help content in the document actions pane 135. For example, if the user places her computer cursor in the "objectives" section 125, the user may be provided with "Objective Writing Tips" 155 shown in the document actions pane 135. Selection of the "Objective Writing Tips" 155, as illustrated in Fig. 3, causes a display of "Objective Writing Tips" text 160 that provide the user with helpful information as to how to complete the "objectives" section the performance review document 110, illustrated in Fig. 3. If the user moves the cursor to a different section of the document, for example the personal information section 120, information provided in the document actions pane 135 will be provided to give the user assistance with the completion of the personal information section 120.

In addition to helpful information to assist the user, a variety of document actions 145 are provided. For example, the "Submit Review" action may allow the user to submit the completed document 110 to her supervisor or to her employee after completion of the document. The "Open Last Review" action may allow the user to open the last performance review so that she may determine whether she completed her objectives as set forth in the last review. If the document in use by the user is some other type of document, for example a resumé document, helpful information in the document actions pane might include information on preparing the "education" section, the "experience" section, and/or the "personal information" section.

According to the present invention, and as described in detail below, when a user focuses on a particular portion of the document 110, such as the "objectives" section of the performance review 110 illustrated in Fig. 3, a solution property 115 points the document to the "objectives" section help solution illustrated in the document actions pane 135. The solution location 118 provides the document 110 and the application 100 with the location of the components, dlls, or XML files necessary for

implementing that solution. As should be understood, exemplary components may include components for drawing the document actions pane 135, components for displaying the information associated with the particular context, in this case the "objectives" section, and components for executing document actions such as "Submit Review" action 145. If the components needed by the application 100 and document 110 to provide the "smart" functionality described above have not been downloaded to the user's computer 20, then these components may be downloaded to the computer 20 in accordance with the present invention as described below.

Fig. 4 is a flow chart illustrating a method for downloading components required for operating and for providing functionality to a client-side software application. As described above, the embodiments of the present invention provide for downloading all components, files, sub-routines, Extensible Markup Language files, and file implementation information necessary for the operation of the software application 100 on the user's computer 20. The embodiments of the present invention also provide for downloading all necessary components, files, XML schema, and associated XML files required by an application 100 for transforming a document 110 into a "smart" document.

The method 400 for downloading required components and files to the user's computer 20, begins at start step 405, and proceeds to step 410 where the application 100 is booted by the user, and the document 110 is opened for editing by the user. According to an embodiment of the present invention, the document 110 opened by the user may be any document that points to required components and files for operation of the document 110 via the application 100, or the document may be a "smart" document that points to a solution that may be downloaded to provide enhanced functionality to the document, as described above with reference to Fig. 3.

At step 415, the application 100 determines whether the document 110 points to or refers to components, sub-routines, or files that are necessary for the operation of the application 100, or whether the document 110 points to a solution that may be downloaded to the user's computer 20 to provide additional functionality to the application 100 for editing the document 110. According to an embodiment of the

present invention, a schema is attached to a document 110 that defines permissible data content, data type and data structure from the document. The document is structured to associate the document with the schema. A solution comprising components and files needed to provide functionality to the application or to transform the application into a "smart" document is associated with the document structure.

According to another embodiment of the invention, an Extensible Markup Language (XML) schema is attached to the document to define permissible data content, data types and data structures for the document. Particular portions of the document are annotated with XML structure associated with the schema. A solution comprising a plurality of software components is associated with XML elements making up the XML structure. The document is enabled to call the solution to obtain functionality provided by the plurality of software components assembled on the manifest 38 upon initiation of editing the document within an XML element associated with a particular solution.

If the document 110 points to various components and files necessary for the operation of the application 100 and for editing the document 110, or if the document points to a solution that may be downloaded to transform the document into a "smart" document, the method proceeds to step 420, and the application 100 checks the schema library 105 resident on the user's computer 20 for the presence of the necessary components or files. If the document includes solution properties pointing to more than one solution, the user of the document may be prompted to select one of the solutions for downloading from the remote server.

Referring back to step 415, if the document does not refer to a solution for transforming the document into a "smart" document, the method may proceed to step 425, where a determination is made as to whether a schema has been attached to the document. If the document has an attached schema, but no reference to a particular solution, the method proceeds to step 420, and the schema library 105 is checked for the presence of components associated with the schema attached to the document 110. If a determination is made that the document does not refer to any required components or files, or that the document does not have an attached schema, the method proceeds to

step 460, and the document is opened in accordance with normal document opening procedures according to the document application 100.

Referring back to step 420, at the schema library 105, a determination is made as to whether the schema library 105 includes files associated with the application 100 and the document 110. A determination as to whether a solution is present in the schema library 105 that is referred to in the document 110 is made by looking for component or solution IDs referred to in the solution properties 115 in the document 110.

At step 430, a determination is made as to whether a download of components, files, sub-routines, or information related thereto should be downloaded to the user's computer 20 to update existing components or to add additional functionality to transform the document 110 into a "smart" document. In order to make the determination as to whether components should be updated, or as to whether additional functionality should be downloaded from the manifest 38, the user may be provided with a prompt, such as the dialog box 600, illustrated in Fig. 6. The dialog box may inform the user that components necessary for the proper operation of the application 100 may have been updated or repaired recently, or that the document she is currently editing will work better if the user downloads components from a given web server site. As shown in the dialog box 600, the user may be provided with a variety of options including acceptance of a download of components, acceptance of automatically downloading future updates, or the user may decline the downloading of updates or components.

A registry key may be written into the computer's registry so that when the application boots, a query can be sent to the manifest 38 via the remote server 49 to determine whether new functionality has been added to the manifest 38 that is associated with the application 100 or with solutions being used by the application 100 to transform the application into a "smart" document. Alternatively, the manifest 38 may include a timing mechanism that contacts the user after a set amount of elapsed time to notify the user to check for updates to files or functionality contained on the manifest 38. The amount of time between checks for updates to the manifest 38 may be set by the creator of the manifest 38 or by the user of the application 100, as described

below. Alternatively, the application 100 can be programmed to call or "ping" the server 49 on some regular frequency to determine whether any software component updates or additions are available. Alternatively, the user may request software component updates to be downloaded from the manifest 38 upon user demand, or the user may choose to have updates downloaded on some regular frequency, such as application boot-up, or daily, weekly, etc. According to an embodiment, the user may be provided with a user interface for choosing the frequency of download from the manifest 38.

At step 435, if the user decides to download updates to existing files or components or new functionality to transform the document 110 into a "smart" document, the user requests the download of the suggested files or functionality, and the application 100 connects to the remote server 49 to request the required files or updates from the manifest 38. In order to connect the application 100 to the remote server 49 to obtain downloaded information from the manifest 38, the application 100 may launch an instance of an Internet browser capable of connecting the user's computer 20 to the remote server 49. Alternatively, the functionality may be integrated into the application 100 for connecting to the remote server 49 in the same manner as an Internet browser would connect to the remote server 49 to download information directly to the application 100 via the user's computer 20. If the user's computer 20 is not adapted for on-line communication, a prompt such as the dialog box 700 illustrated in Fig. 7 may be provided to the user to inform the user that the document may not work properly if the user is off-line.

As should be understood, the call made from the application 100 to the manifest 38 located on the remote server 49 may be made based on a uniform resource locator (URL) designated for obtaining updates to existing files or new functionality and attached to the document at the solution location 118, illustrated in Fig. 1. Alternatively, the user may have a set of URLs from which to obtain updates to software components or additional functionality, and the user may enter one or more of those URLs in response to a prompt informing the user that updating the software components provisioned for her application 100 may be helpful. Alternatively, the user

may decide without being prompted to cause the application 100 to call the manifest 38 to obtain software component updates and additions. According to this embodiment, a function tool may be provided in the application 100 that allows the user to enter the URL of a manifest 38 from which the user desires to obtain software component updates or functionality additions.

Once the user has requested the download of software component updates or functionality additions for use by the application 100, the method proceeds to step 440 and a number of security checks are performed to ensure that the user receives the downloaded files and updates from a trusted source. If the document refers to a manifest 38 via a server site that is on the user's list of trusted sites, or if the manifest is requested from a site on the user's trusted intranet system, then the download of the requested or accepted files or updates may be performed. If the URL at which the manifest 38 is located is not a trusted site or in the user's trusted Intranet system, the user may be notified with a prompt such as the dialog box 800, illustrated in Fig. 8, to notify the user that the site being called is not on the user's list of trusted sites or is not coming via the user's trusted intranet system. The user may then choose to either accept a download of information from the manifest 38 or decline the download in response to the prompt. If the document refers to a manifest 38 that is on a trusted site or that is available via a trusted Intranet system, but where the site containing the manifest 38 is not available, the user may be notified with a prompt such as the dialog box 900 illustrated in Fig. 9 that the site is currently unavailable and that the download of information is likewise not available.

An additional security check that may be performed after the download, but prior to installing components, includes preparing a checksum value of the information provided by the manifest 38 and comparing that to a checksum value prepared for the downloaded information at the user's computer 20. If the two values do not match, the installation to the user's computer 20 is prevented because the variation in the checksum values may be an indication that unauthorized tampering with the contents of the downloaded files occurred in transit to the user's computer 20.

In addition to the foregoing security checks, digital signatures may also be utilized to check the security of files downloaded from the manifest 38. That is, files contained on the manifest 38 may be enhanced with a digital signature of the creator or administrator of those files that may be checked at the remote server 49 or at the user's computer 20 against a list of trusted digital signatures prior to downloading and/or installing information from those files. If any of the foregoing security checks fail to ensure the validity and security of the files contained in the manifest 38, the method proceeds to step 460, and document opening is completed without downloading any additional components or file updates. That is, the document opens and operates without the benefit of new functionality or updates to existing functionality that may be available from the manifest 38. Alternatively, the user may override the security system and accept the downloaded information even if the security check indicates that the information is not coming from a trusted site if the user is otherwise confident that the downloaded information may be trusted.

Referring back to step 445, if the security checks pass, the method proceeds to step 450 where a validation of the contents of the manifest 38 is performed. Before downloading components or information from the manifest to the user's computer 20, a comparison is made between the components and information contained in the manifest 38 (pointed to by the document 110) and the components and information already present in the schema library 105. If the components and information on the manifest 38 does not differ from the components and information already available to the application via the schema library 105, then the download from the manifest 38 may be avoided.

The validation of the manifest 38 includes a determination of which components or information regarding those components are pointed to by the document 110 via the application 100. That is, a solution referred to by the document 110 may require additional functionality, and the solution may point to particular components contained on the manifest 38, but not point to all components contained on the manifest 38. Accordingly, at step 450, a determination is made as to the number of components and

the content of information related to those components that should be downloaded to the user's computer 20 for integration with the application 100.

At step 455, the manifest 38 is processed. Processing the manifest 38 includes actually downloading files, components, subroutines, and related information pointed to by the document 110 or application 100. Once the required files are downloaded to the user's computer 20, those files are installed and registered with the operating system of the computer 20. As should be understood by those skilled in the art, any required registry keys or set-up actions necessary to ensure that the downloaded files properly correspond with the application 100 to provide the required functionality are performed at step 455. After all required files are downloaded from the manifest 38, opening the document 110 is completed with the additional or updated functionality provided by the downloaded file updates or file additions. The method ends at step 490.

Once the document 110 is opened after the download of component updates or additions, the user of the document now has all available functionality downloaded onto the user's computer 20. If the document is a "smart" document as illustrated with reference to Fig. 3, the document will provide the user with all the normal functionality required for operating the document, and additionally, the user will have useful help content and document actions based on the editing context within the document. For example, if the user has opened a resume template, the user may receive help content and document actions based on the position of the user's computer cursor in a section of the document. In addition to pointing to the solution ID for obtaining the solution for use by the document, the document will point to the solution location from where the software components or software component upgrades may be downloaded to upgrade or add to the functionality in use by the document. Accordingly, with the present invention, the user may readily obtain downloaded updates and additions to the functionality available for use with the document being edited.

Moreover, if the user would like to send the document to a second user, there is no need to send the second user an installation package containing all the software components necessary to give the document the enhanced or "smart" functionality. Once the second user opens the document, the second user will be notified that the

document will work more efficiently if the user downloads software components from the manifest 38, as discussed with reference to Fig. 4. If the downloaded information is accepted, the document will then have all the available functionality on the second user's computer 20 that it had on the computer of the first user.

It will be apparent to those skilled in the art that various modifications or variations may be made in the present invention without departing from the scope or spirit of the invention. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein.

WE CLAIM:

1. A method of downloading software components from a remote source to a software application for providing updates or additions to application and document functionality, comprising the steps of:

attaching a schema to a document defining permissible data content, data type and data structure for the document;

structuring the document to associate the document with the schema;

associating a solution with the document structure;

assembling a plurality of software components comprising the solution at a location remote from the document;

enabling the document to call the solution to obtain functionality provided by the plurality of software components; and

downloading the plurality of software components to the application for provision of the functionality to the document.

2. The method of Claim 1, whereby the step of enabling the document to call the solution to obtain functionality provided by the plurality of software components includes:

attaching a solution property to the document for pointing the document to the solution; and

attaching a solution location to the document for enabling the document to locate the solution at the remote location.

3. The method of Claim 2, whereby the solution location includes a uniform resource locator (URL) for the solution.

4. The method of Claim 2, whereby the step of attaching a solution property to the document further comprises attaching a plurality of solution properties to the document to associate a plurality of subsets of the solution with particular portions of the document.

5. The method of Claim 4, further comprising the steps of:

listing the solution and the plurality of software components in a manifest of software components; and

storing the manifest in a remote library of software components on a remote server accessible by the document.

6. The method of Claim 5, further comprising the step of:

attaching a schema to the manifest for defining permissible data content, data type and data structure of the manifest and for associating the solution with the document.

7. The method of Claim 7, prior to the step of downloading the software components to the application, further comprising the steps of:

determining whether the plurality of software components for providing the solution is present in a local library of software components; and

if the plurality of software components is not present in the local library of software components, calling the manifest at the remote library of software components for obtaining the solution.

8. The method of Claim 7, prior to the step of determining whether the plurality of software components for providing the solution is present in a local library of software components, further comprising determining whether the document includes solution properties pointing to more than one solution.

9. The method of Claim 8, whereby if the document includes solution properties pointing to more than one solution, selecting for downloading from the remote server software components associated with one of the more than one solutions.

10. The method of Claim 7, prior to the step of downloading the software components to the application, receiving a query to determine whether the plurality of software components should be downloaded.

11. The method of Claim 10, whereby the step of receiving a query to determine whether the plurality of software components should be downloaded, includes receiving a notification that the plurality of software components have been updated at the remote server.

12. The method of Claim 11, further comprising:

causing the notification to be sent from the manifest upon the occurrence of an update to any of the plurality of software components.

13. The method of Claim 12, further comprising:

causing the notification to be sent from the manifest at a prescribed frequency.

14. The method of Claim 13, whereby the prescribed frequency is set by the creator of the manifest.

15. The method of Claim 11, whereby the step of receiving a query to determine whether the plurality of software components should be downloaded, includes receiving a notification that additional software components have been added to the plurality of software components stored at the remote server.

16. The method of Claim 11, whereby the step of receiving a query to determine whether the plurality of software components should be downloaded, includes receiving a notification that a set amount of time has elapsed since the last download of the plurality of software components to the application.

17. The method of Claim 5, prior to the step of downloading the software components to the application, receiving at the remote server a command from the application to download the software components to the application.

18. The method of Claim 10, whereby the command from the application to download the software components to the application is initiated by a user of the application.

19. The method of Claim 7, whereby prior to the step of downloading the software components to the application, connecting the software application to the remote server.

20. The method of Claim 19, whereby the step of connecting the software application to the remote server includes connecting the software application to the remote server via the Internet.

21. The method of Claim 19, whereby the step of connecting the software application to the remote server includes connecting the software application to the remote server via an intranet.

22. The method of Claim 19, whereby the step of connecting the software application to the remote server includes connecting the software application to the remote server via a distributed computing environment.

23. The method of Claim 22, whereby the distributed computing environment includes a wireline telecommunications network.

24. The method of Claim 22, whereby the distributed computing environment includes a wireless telecommunications network.

25. The method of Claim 19, whereby the step of connecting the software application to the remote server includes connecting the software application to the remote server via a local area network.

26. The method of Claim 7, prior to downloading the software components to the document, checking the security of the manifest.

27. The method of Claim 26, whereby if the security of the manifest cannot be checked, preventing the step of downloading the software components to the document.

28. The method of Claim 27, whereby the step of checking the security of the manifest includes determining whether the manifest is located at a trusted remote site.

29. The method of Claim 27, whereby the step of checking the security of the manifest includes determining whether the manifest is from a trusted intranet.

30. The method of Claim 27, whereby the step of checking the security of the manifest includes determining whether digital signature applied to the manifest matches a trusted digital signature.

31. The method of Claim 30, whereby the digital signature applied to the manifest includes a particular digital signature applied to each of a plurality of files included on the manifest.

32. The method of Claim 19, after the step of connecting the software application to the remote server, further comprising:

obtaining a checksum value representing the contents of the manifest;

comparing the checksum value representing the contents of the manifest with a checksum value representing the contents of the manifest after the contents of the manifest have been received by the application, but before the contents of the manifest have been installed for use by the application; and

if the checksum value representing the contents of the manifest differs from the checksum value representing the manifest after the manifest has been received by the application, preventing the step of downloading the software components to the document.

33. The method of Claim 19, after connecting the software application to the remote server, and before downloading the plurality of software components to the application for provision of the functionality to the document, determining which components of the plurality of components are required by the application to provide a solution pointed to by the solution property.

34. The method of Claim 33, after the step of determining which components of the plurality of components are required by the application, further comprising:

comparing the components present in the manifest that are required by the application with components that are present in the local library of software components;

if the components present in the manifest that are required by the application are in addition to the components present in the local library, downloading the software components to the application; and

if the components present in the manifest that are required by the document are the same as the components that are present in the local library, but the components that are present in the manifest that are required by the document are of a newer version than the components that are present in the local library, downloading the software components to the application.

35. The method of Claim 7, after downloading the software components to the application, installing the downloaded components for use by the application for providing functionality to the document.

36. A method of downloading software components from a remote source to a software application for providing updates or additions to application and document functionality, comprising the steps of:

attaching an Extensible Markup Language (XML) schema to the document defining permissible data content, data types and data structures for the document;

annotating particular portions of the document with XML structure associated with the permissible data content, the permissible data types, and permissible data structures for the particular portions of the document as defined by the schema;

associating a solution with XML elements comprising the XML structure;

assembling a plurality of software components comprising the solution at a location remote from the document;

enabling the document to call the solution to obtain functionality provided by the plurality of software components; and

downloading the plurality of software components to the application for provision of the functionality to the document.

37. The method of Claim 36, whereby the step of enabling the document to call the solution to obtain functionality provided by the plurality of software components includes:

attaching a solution property to the document for pointing the document to the solution; and

attaching a solution location to the document for enabling the document to locate the solution at the remote location.

38. The method of Claim 37, whereby the solution location includes a uniform resource locator (URL) for the solution.

39. The method of Claim 36, whereby the solution comprises a plurality of software components providing help content and document actions to the document via the application.

40. The method of Claim 36, whereby enabling the document to call the solution to obtain functionality provided by the plurality of software components includes enabling the document to call the plurality of software components upon initiation of editing the document within an XML element associated with a particular solution.

41. The method of Claim 37, further comprising the steps of:

listing the solution and the plurality of software components in a manifest of software components; and

storing the manifest in a remote library of software components on a remote server accessible by the document.

42. The method of Claim 41, prior to the step of downloading the software components to the application, querying a user of the document to determine whether the plurality of software components should be downloaded.

43. The method of Claim 41, prior to downloading the software components to the document, further comprising:

determining whether the manifest is located at a trusted remote site; and

if the manifest is not located at a trusted remote site, preventing the step of downloading the plurality of software components to the application.

44. The method of Claim 41, before downloading the plurality of software components to the application for provision of the functionality to the document, further comprising:

connecting the software application to the remote server;

determining which components of the plurality of components are required by the application to provide a solution pointed to by the solution property;

comparing the components present in the manifest that are required by the application with components that are present in the local library of software components;

if the components present in the manifest that are required by the application are in addition to the components present in the local library, downloading the software components to the application; and

if the components present in the manifest that are required by the document are the same as the components that are present in the local library, but the components that are present in the manifest that are required by the document are of a newer version than the components that are present in the local library, downloading the software components to the application.

45. A computer readable medium having stored thereon computer-executable instructions which when executed by a computer, perform the steps of:

attaching an Extensible Markup Language (XML) schema to the document defining permissible data content, data types and data structures for the document;

annotating particular portions of the document with XML structure associated with the permissible data content, the permissible data types, and permissible data structures for the particular portions of the document as defined by the schema;

associating a solution with XML elements comprising the XML structure;

assembling a plurality of software components comprising the solution at a location remote from the document;

enabling the document to call the solution to obtain functionality provided by the plurality of software components; and

downloading the plurality of software components to the application for provision of the functionality to the document.

46. The computer readable medium of Claim 45 having stored thereon computer-executable instructions which when executed by a computer, further perform the step of:

whereby the step of enabling the document to call the solution to obtain functionality provided by the plurality of software components includes attaching a solution property to the document for pointing the document to the solution; and

attaching a solution location to the document for enabling the document to locate the solution at the remote location.

47. The computer readable medium of Claim 46, whereby the solution location includes a uniform resource locator (URL) for the solution.

48. The computer readable medium of Claim 45, whereby the solution comprises a plurality of software components providing help content and document actions to the document via the application.

49. The computer readable medium of Claim 45, whereby enabling the document to call the solution to obtain functionality provided by the plurality of software components includes enabling the document to call the plurality of software components upon initiation of editing the document within an XML element associated with a particular solution.

50. The computer readable medium of Claim 46 having stored thereon computer-executable instructions which when executed by a computer, further perform the step of:

listing the solution and the plurality of software components in a manifest of software components; and

storing the manifest in a remote library of software components on a remote server accessible by the document.

51. The computer readable medium of Claim 49 having stored thereon computer-executable instructions which when executed by a computer, prior to the step of downloading the software components to the application, further performs the steps of:

opening the document;

placing a computer cursor in a particular portion of the document;

generating a list of XML elements enclosing a position of the cursor;

determining whether a solution property pointing the document to the solution is associated with the list of XML elements; and

determining whether the plurality of software components for providing the solution is present in a local library of software components; and

if the plurality of software components is not present in the local library of software components, calling the manifest at the remote library of software components for obtaining the solution.

52. The computer readable medium of Claim 51, whereby the step of determining whether a solution property pointing the document to the solution is associated with the list of XML elements, further comprises the step of parsing a look-up table of solutions to determine if any available help content or document actions are associated with any XML element in the list of XML elements.

53. The computer readable medium of Claim 50, prior to the step of downloading the software components to the application, querying a user of the document to determine whether the plurality of software components should be downloaded.

54. The computer readable medium of Claim 50, prior to downloading the software components to the document, further comprising:

determining whether the manifest is located at a trusted remote site; and

if the manifest is not located at a trusted remote site, preventing the step of downloading the plurality of software components to the application.

55. The computer readable medium of Claim 50, before downloading the plurality of software components to the application for provision of the functionality to the document, further comprising:

connecting the software application to the remote server;

determining which components of the plurality of components are required by the application to provide a solution pointed to by the solution property;

comparing the components present in the manifest that are required by the application with components that are present in the local library of software components;

if the components present in the manifest that are required by the application are in addition to the components present in the local library, downloading the software components to the application; and

if the components present in the manifest that are required by the document are the same as the components that are present in the local library, but the components that are present in the manifest that are required by the document are of a newer version than the components that are present in the local library, downloading the software components to the application.

Abstract

A method and system are provided for downloading software components from a remote source to a software application for providing updates or additions to the application's functionality. All components and files that may be utilized to update or add to functionality available to the application are identified and are assembled on a manifest that may be located on a remote web server accessible by the application. If components of the application should be updated, or corrections or improvements to existing components are available, or new functionality is available that will transform the user's existing application and document into a "smart" application and "smart" document, the application may connect to the web server to download the required components.

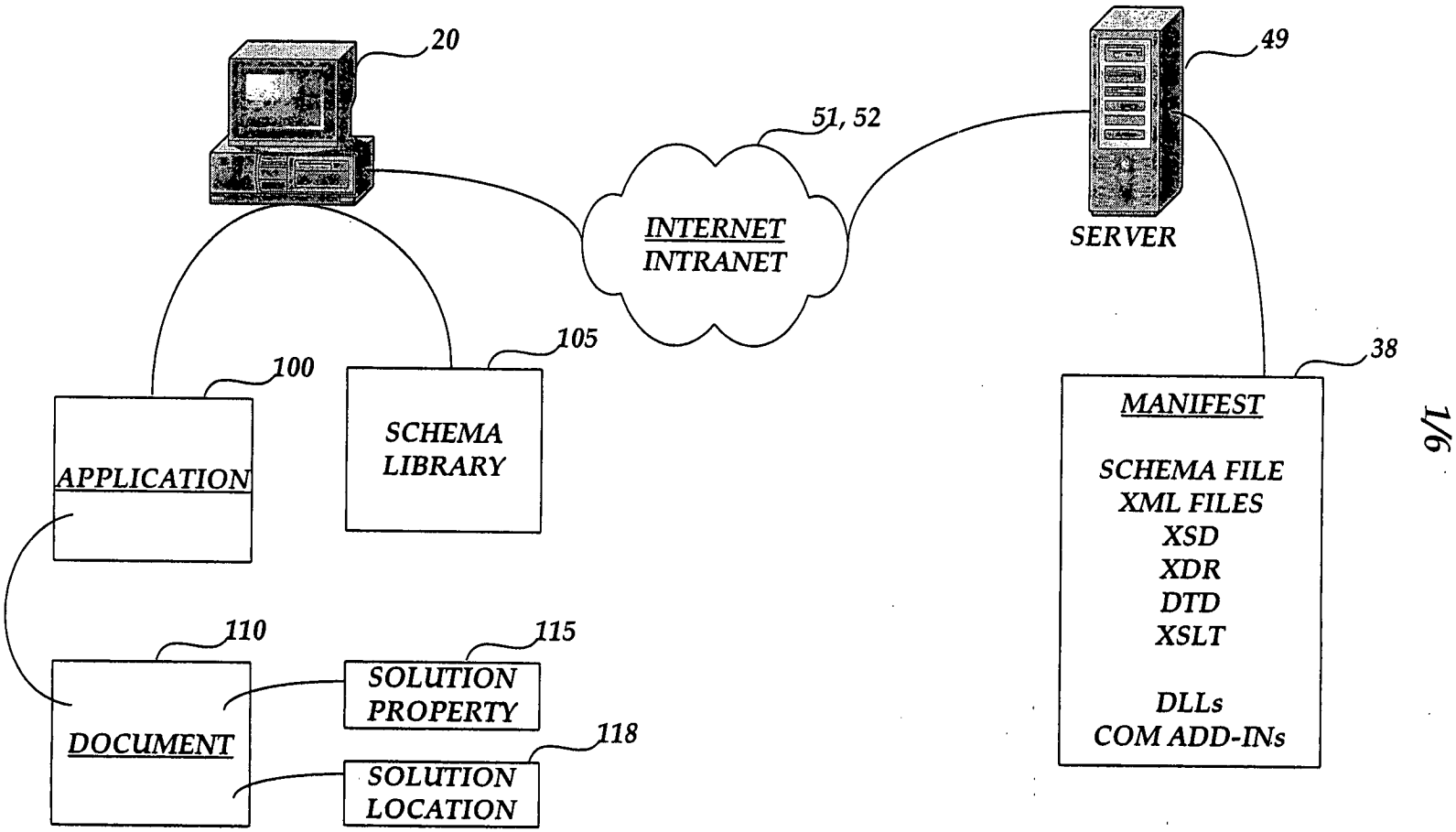
20
49
51, 52
INTERNET
INTRANET
SERVER
105
100
38
SCHEMA
LIBRARY
APPLICATION
MANIFEST
SCHEMA FILE
XML FILES
XSD
XDR
DTD
XSLT
DLLs
COM ADD-INs
110
115
SOLUTION
PROPERTY
DOCUMENT
118
SOLUTION
LOCATION

Fig. 1

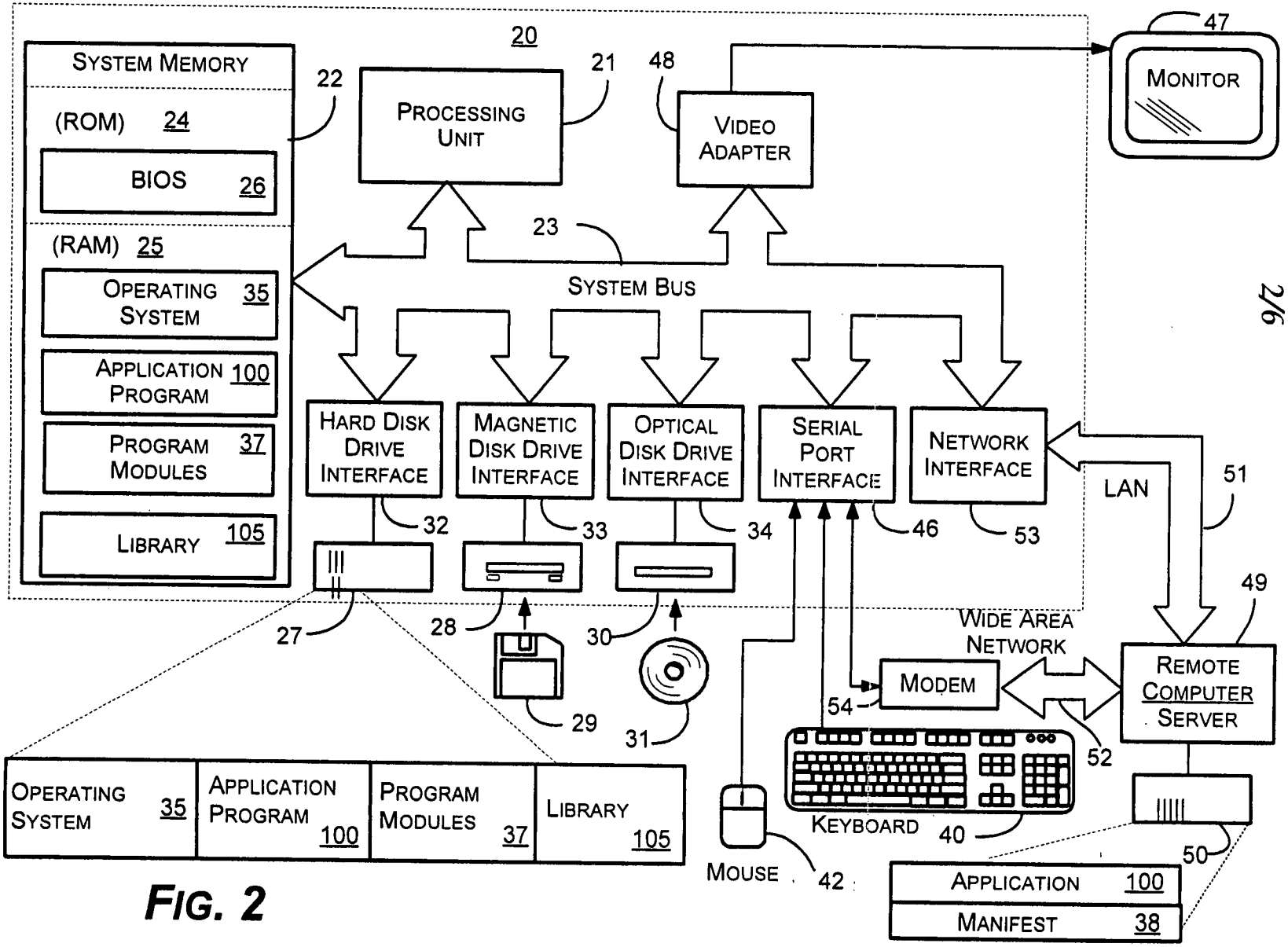
20
SYSTEM MEMORY
22
(ROM) 24
BIOS 26
(RAM) 25
OPERATING SYSTEM 35
APPLICATION PROGRAM 100
PROGRAM MODULES 37
LIBRARY 105
PROCESSING UNIT
21
48
VIDEO ADAPTER
47
MONITOR
23
SYSTEM BUS
HARD DISK DRIVE INTERFACE
MAGNETIC DISK DRIVE INTERFACE
OPTICAL DISK DRIVE INTERFACE
SERIAL PORT INTERFACE
NETWORK INTERFACE
32
33
34
46
53
LAN
51
27
28
30
29
31
49
WIDE AREA NETWORK
MODEM
54
REMOTE COMPUTER SERVER
52
KEYBOARD 40
MOUSE 42
50
APPLICATION 100
MANIFEST 38
OPERATING SYSTEM 35
APPLICATION PROGRAM 100
PROGRAM MODULES 37
LIBRARY 105

FIG. 2

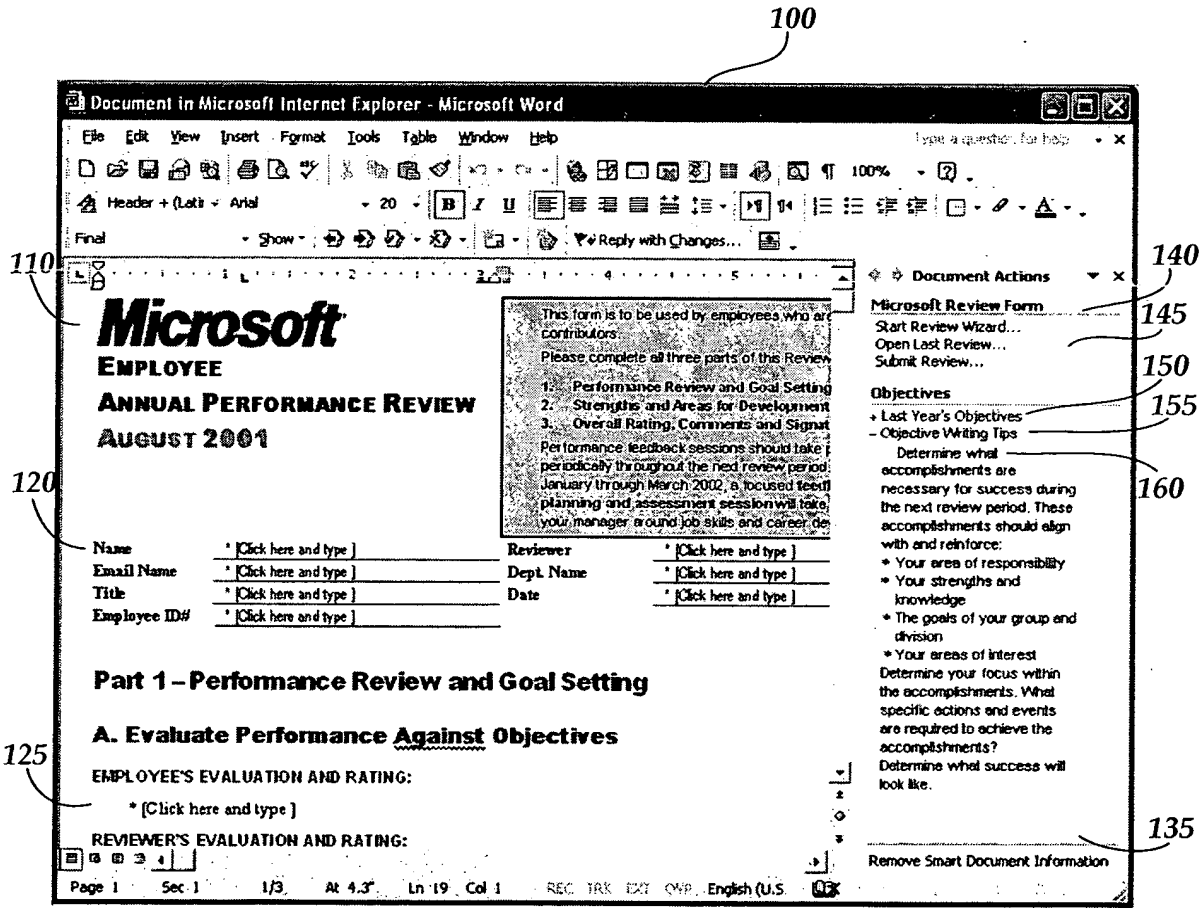
100
Document in Microsoft Internet Explorer - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Header + (Lati Arial 20
Final Show Reply with Changes...
110
Microsoft
EMPLOYEE
ANNUAL PERFORMANCE REVIEW
AUGUST 2001
This form is to be used by employees who ar
contributors.
Please complete all three parts of this Review
1. Performance Review and Goal Setting
2. Strengths and Areas for Development
3. Overall Rating, Comments and Signat
Performance feedback sessions should take p
periodically throughout the next review period
January through March 2002, a focused feedb
planning and assessment session will take
your manager around job skills and career de
120
Name * [Click here and type]
Email Name * [Click here and type]
Title * [Click here and type]
Employee ID# * [Click here and type]
Reviewer * [Click here and type]
Dept. Name * [Click here and type]
Date * [Click here and type]
Part 1 – Performance Review and Goal Setting
A. Evaluate Performance Against Objectives
125
EMPLOYEE'S EVALUATION AND RATING:
* [Click here and type]
REVIEWER'S EVALUATION AND RATING:
Page 1 Sec 1 1/3 At 4.3" Ln 19 Col 1 REC TRK EXT OVR English (U.S.
140
Document Actions
Microsoft Review Form
145
Start Review Wizard...
Open Last Review...
Submit Review...
150
Objectives
+ Last Year's Objectives
155
- Objective Writing Tips
Determine what accomplishments are necessary for success during the next review period. These accomplishments should align with and reinforce:
160
* Your area of responsibility
* Your strengths and knowledge
* The goals of your group and division
* Your areas of interest
Determine your focus within the accomplishments. What specific actions and events are required to achieve the accomplishments?
Determine what success will look like.
135
Remove Smart Document Information

Fig. 3

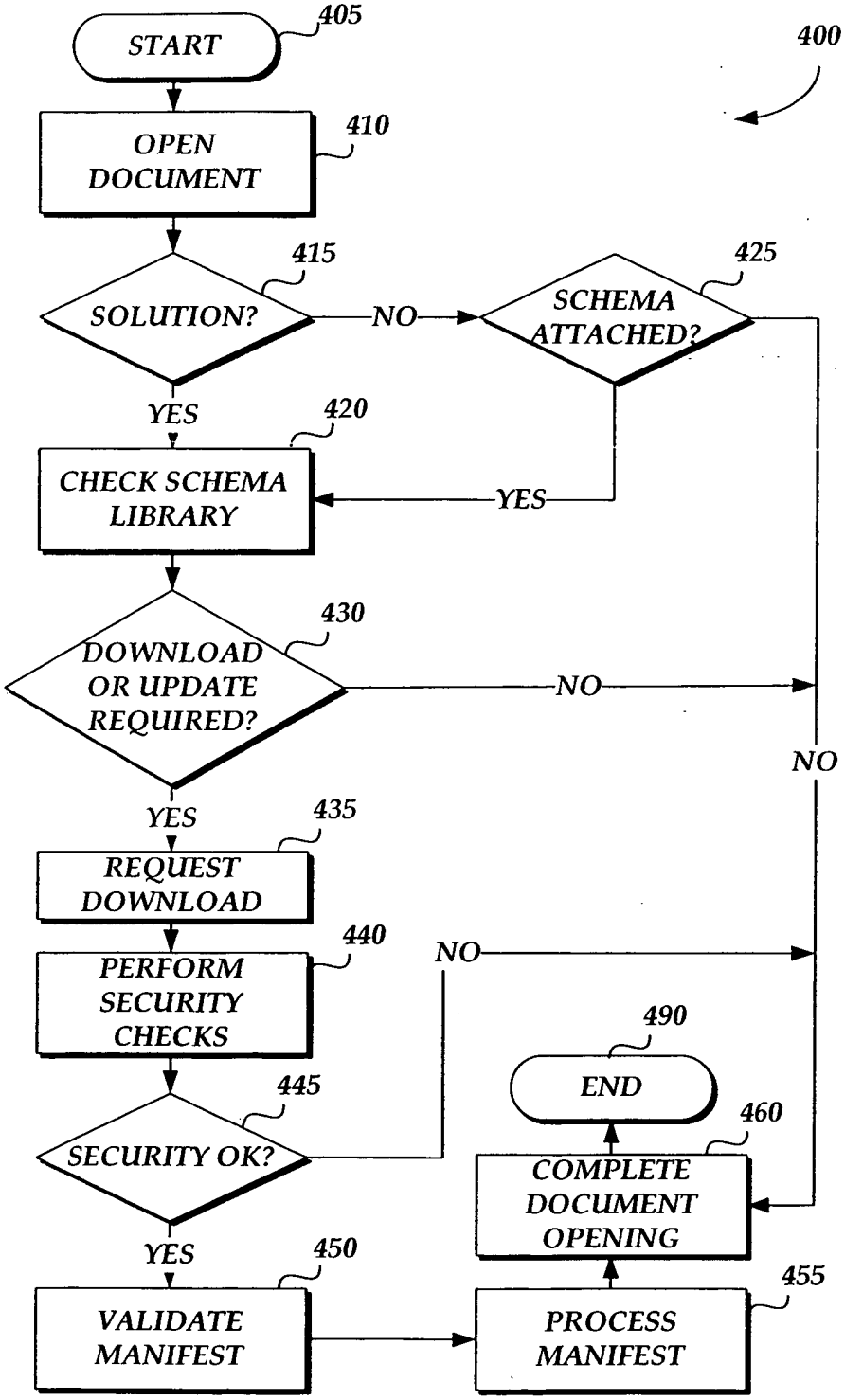
START
405
400
OPEN DOCUMENT
410
SOLUTION?
415
NO
SCHEMA ATTACHED?
425
YES
420
CHECK SCHEMA LIBRARY
YES
DOWNLOAD OR UPDATE REQUIRED?
430
NO
NO
YES
435
REQUEST DOWNLOAD
PERFORM SECURITY CHECKS
440
NO
490
END
SECURITY OK?
445
460
COMPLETE DOCUMENT OPENING
YES
450
455
VALIDATE MANIFEST
PROCESS MANIFEST

Fig. 4

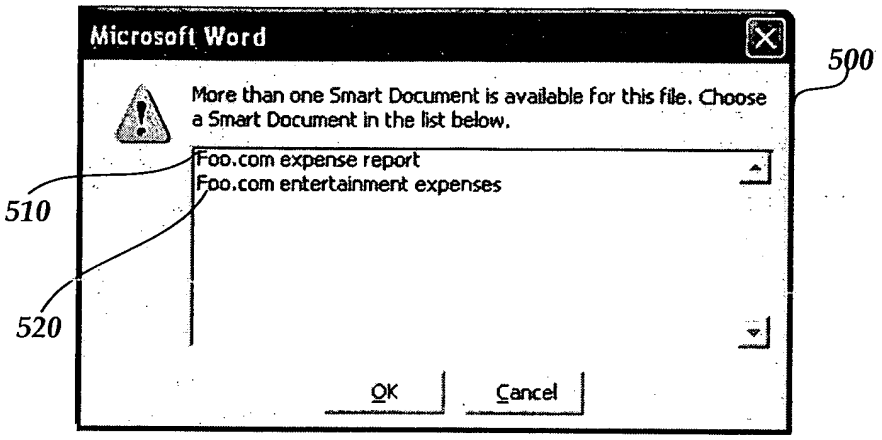
Microsoft Word
More than one Smart Document is available for this file. Choose a Smart Document in the list below.
Foo.com expense report
Foo.com entertainment expenses
OK
Cancel
500
510
520

Fig. 5

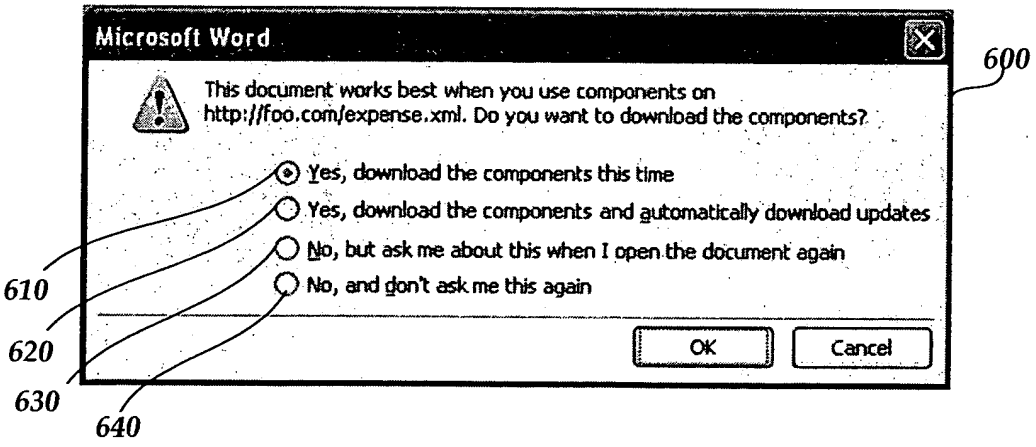
Microsoft Word
This document works best when you use components on http://foo.com/expense.xml. Do you want to download the components?
Yes, download the components this time
Yes, download the components and automatically download updates
No, but ask me about this when I open the document again
No, and don't ask me this again
OK
Cancel
600
610
620
630
640

Fig. 6

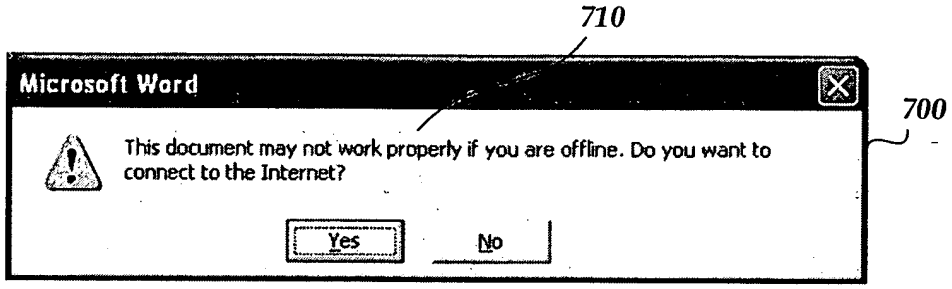

Fig. 7

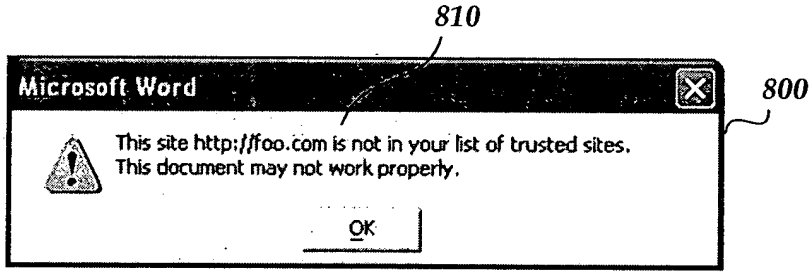

Fig. 8

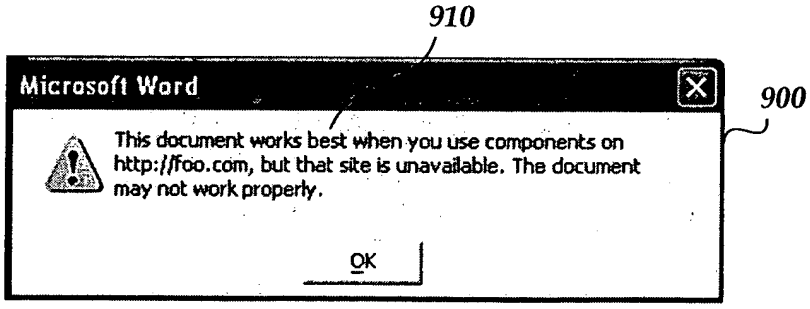

Fig. 9